Practical Uniform Interpolation and Forgetting for $\mathcal{ALC}$ TBoxes with Applications to Logical Difference *

Michel Ludwig
Institute for Theoretical Computer Science
TU Dresden, Germany
michel@tcs.inf.tu-dresden.de

Boris Konev
Department of Computer Science
University of Liverpool, United Kingdom
konev@liverpool.ac.uk

Abstract

We develop a clausal resolution-based approach for computing uniform interpolants of TBoxes formulated in the description logic $\mathcal{ALC}$ when such uniform interpolants exist. We also present an experimental evaluation of our approach and of its application to the logical difference problem for real-life $\mathcal{ALC}$ ontologies. Our results indicate that in many practical cases uniform interpolants exist and that they can be computed with the presented algorithm.

Introduction

Ontologies or *TBoxes* expressed in Description Logics (DL) provide a common vocabulary for a domain of interest together with a description of the meaning of the terms built from the vocabulary and of the relationships between them. Modern applications of ontologies, especially in the biological, medical, or healthcare domain, often demand large and complex ontologies; for example, the National Cancer Institute ontology (NCI) consists of more than 60 000 term definitions. For developing, maintaining, and deploying such large-scale ontologies it can be advantageous for ontology engineers to concentrate on specific parts of an ontology and ignore or *forget* the rest. Ignoring parts of an ontology can be formalised with the help of *predicate forgetting* and its dual *uniform interpolation*, which have both been extensively studied in the AI and DL literature (ten Cate et al. 2006; Eiter et al. 2006; Herzig and Mengin 2008; Konev, Walther, and Wolter 2009; Wang et al. 2008; 2010; 2012; Lutz and Wolter 2011).

Forgetting parts of an ontology can be used, for example, in the following practical scenarios. *Exhibiting hidden relations*: in addition to the explicitly stated connections between terms, additional relations can also be derived from ontologies with the help of reasoners. Such inferred connections are often harder to understand or debug. By forgetting everything but a handful of terms of interest, it then becomes possible to exhibit inferred relations that were hidden initially, potentially simplifying the understanding of the ontology structure. *Ontology obfuscation*: in software engineering, obfuscation (Collberg, Thomborson, and Low 1998) transforms a given program into a functionally equivalent one that is more difficult to read and understand for humans for the purpose of preventing reverse engineering. Forgetting can provide a similar function in the context of ontology engineering. Terms are often defined with the help of auxiliary terms which give structure to TBox inclusions. However, such a structure might be considered proprietary knowledge that should not be exposed, or it could simply be of little interest for ontology users. By forgetting these intermediate auxiliary terms, we obtain an ontology that is functionally equivalent, yet harder to read, understand, and modify by humans. Further applications of forgetting can be found in (Konev, Walther, and Wolter 2009; Lutz, Seylan, and Wolter 2012).

A promising and important application area of forgetting is the computation of the *logical difference* between ontology versions. Determining whether two versions of a document have differences is a standard task in information technology, and finding differences is particularly relevant for text processing and software development. Already in these areas, it is important to be able to identify which changes are *significant* and which are not (e.g., a software developer might want to ignore changes in the formatting style of the code such as the number of indentation spaces). Detecting significant changes is even more important in the setting of Knowledge Representation, where differences in the knowledge captured by ontologies are often more relevant than syntactic changes. Arguably, one of the most important concerns of an ontology engineer when modifying an existing ontology is to ensure that the introduced changes do not interfere with the meaning of the terms outside the fragment under consideration. Notice that neither the version comparison based on the syntactic form of the documents representing ontologies (Conradi and Westfechtel 1998) nor methods based on the structural transformations of ontology statements (Noy and Musen 2002; Klein et al. 2002; Jiménez-Ruiz et al. 2011) can be used to identify changes to the logical meaning of terms in every situation. However, such a correctness guarantee can be achieved by checking the equivalence of the ontologies resulting from forgetting the terms under consideration before and after the changes occurred.

*Partially supported by the DFG within the Cluster of Excellence 'Center for Advancing Electronics Dresden' (cfAED) and by the EPSRC under the grant EP/H043594/1.

In this paper we develop an algorithm based on clausal resolution for computing uniform interpolants of TBoxes formulated in the description logic $\mathcal{ALC}$ which can preserve all the consequences which do not make use of some given *concept names* and which have a quantifier nesting level limited to some given depth. Subsequently, we present an experimental evaluation of our approach which demonstrates that in many practical cases uniform interpolants (without limitations on the quantifier nesting depth) exist and that they can be computed with our algorithm. We also apply our prototype tool to compute the logical difference between versions of ontologies from the biomedical domain.

The material presented here extends earlier work (Ludwig and Konev 2013). All the missing proofs can be found in the full version of this paper.[1]

Related work. Until recently research on uniform interpolation and forgetting in the setting of DL mainly has concentrated on theoretical foundations of forgetting. This could be partly explained by the high computational complexity of this task and by the fact that uniform interpolants do not always exist. The notion of forgetting has been introduced by Reiter and Lin (1994). Konev, Walther, and Wolter (2009) prove tractability of uniform interpolation for $\mathcal{EL}$ TBoxes of a specific syntactic form. Uniform interpolation for general $\mathcal{EL}$-TBoxes has been investigated by Nikitina and Rudolph (2012).

Wang et al. (2008; 2010; 2012) have developed algorithms for forgetting in expressive description logics. A tight 2-ExpTime-complete bound on the complexity for deciding the existence of a Σ-uniform interpolant in $\mathcal{ALC}$ and a worst-case triple-exponential procedure for computing a Σ-uniform interpolant if it exists, have been given by Lutz and Wolter (2011). Koopmann and Schmidt (2013) have introduced a two-stage resolution-based algorithm for computing uniform interpolants. As outcome of the first stage, a representation of the uniform interpolant in a description logic with fixpoint operators is computed (such a representation always exists) Then in the second stage an attempt is made to eliminate the newly-introduced fixpoints (which may not succeed). In contrast to this approach, our algorithm has one stage and it can be guaranteed that a uniform interpolant of bounded depth is returned.

The notion of the logical difference has been introduced by Konev, Walther, and Wolter (2008) as a way of capturing the difference in the meaning of terms that is independent of the representation of ontologies.

Preliminaries

We start with introducing the description logic $\mathcal{ALC}$. Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countably infinite and mutually disjoint sets of *concept names* and *role names*. *$\mathcal{ALC}$-concepts* are built according to the following syntax rule

$$C ::= \quad A \quad | \quad \top \quad | \quad \neg C \quad | \quad \exists r.C \quad | \quad C \sqcap D,$$

where $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. As usual, other $\mathcal{ALC}$ concept constructors are introduced as abbreviations: $\bot$ stands for $\neg\top$, $C \sqcup D$ stands for $\neg(\neg C \sqcap \neg D)$ and $\forall r.C$ stands for $\neg\exists r.\neg C$. An $\mathcal{ALC}$-TBox $\mathcal{T}$ is a finite set of $\mathcal{ALC}$-inclusions of the form $C \sqsubseteq D$, where C and D are $\mathcal{ALC}$-concepts. A concept equation $C \equiv D$ is an abbreviation for the two inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$. An $\mathcal{ALC}$-TBox $\mathcal{T}$ is *acyclic* if all its inclusions are of the form $A \sqsubseteq C$ and $A \equiv C$, where $A \in \mathsf{N_C}$ and C is an $\mathcal{ALC}$-concept, such that no concept name occurs more than once on the left-hand side and $\mathcal{T}$ contains no cycle in its definitions, i.e., it does not contain inclusions $A_1 \bowtie C_1, \ldots, A_k \bowtie C_k$, where $\bowtie \in \{\sqsubseteq, \equiv\}$, such that A_{i+1} occurs in C_i, for $i = 1, \ldots, k-1$ and $A_k = A_1$.

A signature Σ is a finite subset of $\mathsf{N_C} \cup \mathsf{N_R}$. The signature of a concept C, denoted by $\mathsf{sig}(C)$, is the set of concept and role names that occur in C. If $\mathsf{sig}(C) \subseteq \Sigma$, we call C a Σ-concept. We assume that the two previous definitions also apply to concept inclusions/equations $C \bowtie D$ with $\bowtie \in \{\sqsubseteq, \equiv\}$ and to TBoxes $\mathcal{T}$. The size of a concept C is the length of the string that represents it, where concept names and role names are considered to be of length one. The size of an inclusion/equation $C \bowtie D$ with $\bowtie \in \{\sqsubseteq, \equiv\}$ is the sum of the sizes of C and D plus one. The size of a TBox $\mathcal{T}$ is the sum of the sizes of its inclusions.

The semantics of $\mathcal{ALC}$ is given by *interpretations* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function mapping each concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The *extension* $C^{\mathcal{I}}$ of a concept C is defined by induction as follows:

$$\begin{array}{lcl} \top^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} & := & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (\exists r.C)^{\mathcal{I}} & := & \{\, d \in \Delta^{\mathcal{I}} \mid \exists\, e \in C^{\mathcal{I}} : (d, e) \in r^{\mathcal{I}} \,\} \\ (C \sqcap D)^{\mathcal{I}} & := & C^{\mathcal{I}} \cap D^{\mathcal{I}}. \end{array}$$

Then $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$, in symbols $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

We say that an interpretation $\mathcal{I}$ is a *model of a TBox* $\mathcal{T}$ if $\mathcal{I} \models C \sqsubseteq D$ for all $C \sqsubseteq D \in \mathcal{T}$. An $\mathcal{ALC}$-inclusion $C \sqsubseteq D$ *follows from* (or is *entailed by*) a TBox $\mathcal{T}$ if every model of $\mathcal{T}$ is a model of $C \sqsubseteq D$, in symbols $\mathcal{T} \models C \sqsubseteq D$. We use $\models C \sqsubseteq D$ to denote that $C \sqsubseteq D$ follows from the empty TBox. Finally, a TBox $\mathcal{T}'$ *follows from* (or is *entailed by*) a TBox $\mathcal{T}$ if every model of $\mathcal{T}$ is a model of $\mathcal{T}'$, in symbols $\mathcal{T} \models \mathcal{T}'$.

We now introduce the main notion that we study in this paper.

Definition 1. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ be a signature. We say that an $\mathcal{ALC}$-TBox $\mathcal{T}_\Sigma$ is a* Σ-uniform interpolant *of the TBox $\mathcal{T}$ iff $\mathsf{sig}(\mathcal{T}_\Sigma) \subseteq \Sigma$, $\mathcal{T} \models \mathcal{T}_\Sigma$, and for every $\mathcal{ALC}$ Σ-concept inclusion $C \sqsubseteq D$ with $\mathcal{T} \models C \sqsubseteq D$ it holds that $\mathcal{T}_\Sigma \models C \sqsubseteq D$.*

Uniform interpolation can be seen as the dual notion of *forgetting*: a TBox $\mathcal{T}_\Upsilon$ is the result of forgetting about a signature Υ in a TBox $\mathcal{T}$ iff $\mathcal{T}_\Upsilon$ is a uniform interpolant of $\mathcal{T}$ w.r.t. $\Sigma = \mathsf{sig}(\mathcal{T}) \setminus \Upsilon$. As the following example shows, uniform interpolants of $\mathcal{ALC}$-TBoxes do not always exist.

Example 2. *Let $\mathcal{T} = \{A \sqsubseteq B,\ B \sqsubseteq C \sqcap \exists r.B\}$ and $\Sigma = \{A, C, r\}$. Then there does not exist a Σ-uniform in-*

[1] Available from `http://lat.inf.tu-dresden.de/~michel/publications/`

terpolant of $\mathcal{T}$ as (in particular) the infinite number of consequences of the form $A \sqsubseteq \exists r.C$, $A \sqsubseteq \exists r.\exists r.C, \ldots$ cannot be captured by an $\mathcal{ALC}$-TBox $\mathcal{T}'$ with $\mathsf{sig}(\mathcal{T}') \subseteq \Sigma$. On the other hand, for $\mathcal{T}' = \{A \sqsubseteq B,\ B \sqsubseteq C \sqcap \exists r.B,\ D \equiv B\}$ and $\Sigma' = \{A, C, D, r\}$, a Σ'-uniform interpolant of $\mathcal{T}'$ is $\{A \sqsubseteq D,\ D \sqsubseteq C \sqcap \exists r.D\}$.

Uniform interpolation is also related to the notion of logical difference between ontologies.

Definition 3. *The Σ-logical difference between $\mathcal{ALC}$-TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ is the set $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ of all $\mathcal{ALC}$-concept inclusions $C \sqsubseteq D$ such that $\mathsf{sig}(C \sqsubseteq D) \subseteq \Sigma$, $\mathcal{T}_1 \models C \sqsubseteq D$, and $\mathcal{T}_2 \not\models C \sqsubseteq D$.*

It is easy to see that $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2) = \emptyset$ if, and only if, $\mathcal{T}_2 \models \mathcal{T}_1^{(\Sigma)}$ where $\mathcal{T}_1^{(\Sigma)}$ is a Σ-uniform interpolant of $\mathcal{T}_1$. Moreover, if $\mathcal{T}_2 \not\models \mathcal{T}_1^{(\Sigma)}$, every inclusion $C \sqsubseteq D \in \mathcal{T}_1^{(\Sigma)}$ with $\mathcal{T}_2 \not\models C \sqsubseteq D$ can be regarded as a *witness* of $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$.

With the exception of acyclic $\mathcal{EL}$-TBoxes, checking whether the logical difference between two ontologies is nonempty is at least one exponential harder than reasoning (Konev et al. 2012). Additionally, if the set $\mathsf{Diff}_\Sigma(\mathcal{T}_1, \mathcal{T}_2)$ is nonempty, it is typically infinite. Therefore, in practice, the notion of logical difference is primarily used as a theoretical underpinning of its *approximations* that limit the choice of inclusions $C \sqsubseteq D$ in Definition 3 to Σ-inclusions constructed according to some syntactic rules, see e.g. (Jiménez-Ruiz et al. 2009), (Gonçalves, Parsia, and Sattler 2012).

Computing Uniform Interpolants by $\mathcal{ALC}$-Resolution

The aim of our work is to investigate a practical approach for computing uniform interpolants when they exist. Note that the worst-case optimal procedure given by Lutz and Wolter (2011) is inherently inefficient as it requires one to explicitly construct the internalisation $C_\mathcal{T}$ of a given TBox $\mathcal{T}$, i.e. a concept $C_\mathcal{T}$ of size double exponential in the size of $\mathcal{T}$ having the property that for any inclusion $C \sqsubseteq D$ it holds that $\mathcal{T} \models C \sqsubseteq D$ iff $\models C \sqcap C_\mathcal{T} \sqsubseteq D$.

Our approach is to introduce a resolution-like calculus for $\mathcal{ALC}$ that derives consequences of a TBox $\mathcal{T}$ such that a concept inclusion $C \sqsubseteq D$ is entailed by $\mathcal{T}$ iff a contradiction can be derived from $\mathcal{T}$ and $C \sqcap \neg D$. Similarly to Herzig and Mengin (2008), we then show that any derivation can be restructured in such a way that inferences on selected concept names always precede inferences on other concept names. Then, if the signature Σ is such that $\mathsf{sig}(\mathcal{T}) \setminus \Sigma$ only contains concept names, we generate a set of Σ-consequences $\mathcal{T}'$ of $\mathcal{T}$ by applying the inference rules in a forward chaining manner such that for an arbitrary Σ-inclusion $C \sqsubseteq D$ a contradiction can be derived from $\mathcal{T}$ and $C \sqcap \neg D$ iff a contradiction can be derived from $\mathcal{T}'$ and $C \sqcap \neg D$. Thus, if the forward-chaining process terminates, $\mathcal{T}'$ is a Σ-uniform-interpolant for $\mathcal{T}$.

$\mathcal{ALC}$-Resolution. $\mathcal{ALC}$-resolution operates on $\mathcal{ALC}$ formulae in a conjunctive normal form that is defined according to the following grammar (which is similar to the normal form introduced by Herzig and Mengin (2008)):

$$\begin{array}{rcl} \text{Literal} & ::= & A \mid \neg A \mid \forall r.\text{Clause} \mid \exists r.\text{CNF} \\ \text{Clause} & ::= & \text{Literal} \mid \text{Clause} \sqcup \text{Clause} \mid \bot \\ \text{CNF} & ::= & \top \mid \text{Clause} \mid \text{Clause} \sqcap \text{CNF} \end{array}$$

To simplify the presentation, we assume that clauses are sets of literals and that CNF expressions are sets of clauses. Then $\bot$ corresponds to the empty clause and $\top$ to the empty set of clauses. In the following, the calligraphic letters $\mathcal{C}, \mathcal{D}, \mathcal{E}$ symbolise clauses and $\mathcal{F}, \mathcal{G}$ represent sets of clauses. Similarly to first-order formulae, every $\mathcal{ALC}$ concept can be transformed into an equivalent set of $\mathcal{ALC}$ clauses. The depth of a clause $\mathcal{C}$, $\mathrm{Depth}(\mathcal{C})$, is defined to be the maximal nesting depth of the quantifiers contained in $\mathcal{C}$.

We additionally assume that every clause is assigned a *type*. Clauses obtained from the clausification of TBox inclusions are of the type *universal*, and clauses resulting from the clausification of inclusions to be tested for entailment by the TBox are of the type *initial*. The type of a derived clause is determined by the types of the clauses from which it is derived and by the derivation rule that is used.

Example 4. *The clausification of $\mathcal{T}$ from Example 2 produces three universal clauses: $\neg A \sqcup B$, $\neg B \sqcup C$, $\neg B \sqcup \exists r.B$.*

We now introduce the two resolution calculi $\mathfrak{T}$ and $\mathfrak{T}^u$. The former calculus assumes the TBox to be empty, whereas the latter takes TBox inclusions into account. Thus, $\mathfrak{T}$ derives the empty clause from the set of initial clauses stemming from the clausification of an inclusion $\top \sqsubseteq C \sqcap \neg D$ iff $\models C \sqsubseteq D$; and $\mathfrak{T}^u$ derives the empty clause from the universal clauses stemming from the clausification of a TBox $\mathcal{T}$ and the initial clauses stemming from the clausification of an inclusion $\top \sqsubseteq C \sqcap \neg D$ iff $\mathcal{T} \models C \sqsubseteq D$.

The calculus $\mathfrak{T}$ is defined with the help of the relation $\Rightarrow_\alpha$ given in Fig. 1. For every $\alpha \in \mathsf{N_C} \cup \{\bot\}$, the relation $\Rightarrow_\alpha$ associates with a set of clauses $\mathcal{N}$ a new clause $\mathcal{C}$ which can be 'derived' from the set $\mathcal{N}$ by 'resolving' on α. $\mathfrak{T}$ now consists of the following two inference rules.

$$\frac{\mathcal{C}}{\mathcal{E}}\ (\text{if } \mathcal{C} \Rightarrow_\alpha \mathcal{E}) \qquad\qquad \frac{\mathcal{C} \quad \mathcal{D}}{\mathcal{E}}\ (\text{if } \mathcal{C}, \mathcal{D} \Rightarrow_\alpha \mathcal{E}),$$

where $\mathcal{C}, \mathcal{D}$, and $\mathcal{E}$ are initial clauses.

The calculus $\mathfrak{T}^u$ operates initial and universal clauses and also consists of two rules:

$$\frac{\mathcal{C}}{\mathcal{E}}\ (\text{if } \mathcal{C} \Rightarrow_\alpha \mathcal{E}) \qquad\qquad \frac{\mathcal{C}' \quad \mathcal{D}}{\mathcal{E}'}\ (\text{if } \mathcal{C}', \mathcal{D} \Rightarrow^u_\alpha \mathcal{E}'),$$

where $\mathcal{C}, \mathcal{C}', \mathcal{D}$ are initial or universal clauses, and $\mathcal{C}', \mathcal{D} \Rightarrow^u_\alpha \mathcal{E}'$ holds iff either $\mathcal{C}', \mathcal{D} \Rightarrow_\alpha \mathcal{E}'$, or $\mathcal{D}$ is a universal clause and there exist role names $r_1, \ldots, r_n \in \mathsf{N_R}$ ($n \geq 1$) such that $\mathcal{C}', \forall r_1. \ldots \forall r_n.\mathcal{D} \Rightarrow_\alpha \mathcal{E}'$. (Intuitively, the calculus $\mathfrak{T}^u$ allows for inferences with universal clauses at arbitrary nesting levels of quantifiers, which the calculus $\mathfrak{T}$ does not.) Then $\mathcal{E}$ is a universal clause if $\mathcal{C}$ is a universal clause, and an initial clause otherwise. Similarly, $\mathcal{E}'$ is a universal clause if both $\mathcal{C}'$ and $\mathcal{D}$ are universal clauses, and an initial clause otherwise.

We assume that every clause $\mathcal{E}$ that results from a $\mathfrak{T}$- or $\mathfrak{T}^u$-inference is implicitly simplified by exhaustively removing all occurrences of literals of the form $\exists r.(\mathcal{F}, \bot)$.

(rule $\bot$)	$\mathcal{C}_1' \sqcup \forall r.\bot,\ \ \mathcal{C}_2' \sqcup \exists r.\mathcal{F} \Longrightarrow_\bot \mathcal{C}_1' \sqcup \mathcal{C}_2'$	
(rule A)	$\mathcal{C}_1' \sqcup A,\ \ \mathcal{C}_2' \sqcup \neg A \Longrightarrow_A \mathcal{C}_1' \sqcup \mathcal{C}_2'$	
(rule $\forall\exists$)	$\mathcal{C}_1' \sqcup \forall r.\mathcal{C}_1,\ \ \mathcal{C}_2' \sqcup \exists r.(\mathcal{C}_2, \mathcal{F}) \Longrightarrow_\alpha \mathcal{C}_1' \sqcup \mathcal{C}_2' \sqcup \exists r.(\mathcal{C}_2, \mathcal{F}, \mathcal{C}_3),$	if $\mathcal{C}_1, \mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_3$
(rule $\forall\forall$)	$\mathcal{C}_1' \sqcup \forall r.\mathcal{C}_1,\ \ \mathcal{C}_2' \sqcup \forall r.\mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_1' \sqcup \mathcal{C}_2' \sqcup \forall r.\mathcal{C}_3,$	if $\mathcal{C}_1, \mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_3$
(rule $\exists_1$)	$\mathcal{C}' \sqcup \exists r.(\mathcal{C}_1, \mathcal{F}) \Longrightarrow_\alpha \mathcal{C}' \sqcup \exists r.(\mathcal{C}_1, \mathcal{F}, \mathcal{C}_2),$	if $\mathcal{C}_1 \Longrightarrow_\alpha \mathcal{C}_2$
(rule $\exists_2$)	$\mathcal{C}' \sqcup \exists r.(\mathcal{C}_1, \mathcal{C}_2, \mathcal{F}) \Longrightarrow_\alpha \mathcal{C}' \sqcup \exists r.(\mathcal{C}_1, \mathcal{C}_2, \mathcal{F}, \mathcal{C}_3),$	if $\mathcal{C}_1, \mathcal{C}_2 \Longrightarrow_\alpha \mathcal{C}_3$
(rule $\forall$)	$\mathcal{C}' \sqcup \forall r.\mathcal{C}_1 \Longrightarrow_\alpha \mathcal{C}' \sqcup \forall r.\mathcal{C}_2,$	if $\mathcal{C}_1 \Longrightarrow_\alpha \mathcal{C}_2$

Figure 1: Rules of $\Longrightarrow_\alpha$.

Example 5. *For the universal clauses from Example 4, we have for instance,*

$$\neg A \sqcup B, \neg B \sqcup \exists r.B \ \Rightarrow_B \ \neg A \sqcup \exists r.B \quad \textit{by (rule A).}$$

So, the universal clause $\neg A \sqcup \exists r.B$ is derivable by $\mathfrak{T}^u$ from $\neg A \sqcup B$ and $\neg B \sqcup \exists r.B$. As $\neg B \sqcup C$ is a universal clause and

$$\neg B \sqcup \exists r.B, \forall r.\neg B \sqcup C \Rightarrow_B \neg B \sqcup \exists r.(B, C) \quad \textit{by (rule } \forall\exists\textit{),}$$

the universal clause $\neg B \sqcup \exists r.(B, C)$ is derivable by $\mathfrak{T}^u$ from $\neg B \sqcup \exists r.B$ and $\neg B \sqcup C$. By applying the inference rules to old and newly generated clauses, one can conclude that the universal clauses $\neg A \sqcup \exists r.(B, C)$ and $\neg A \sqcup \exists r.(B, \exists r.B)$ are also derivable by $\mathfrak{T}^u$ from $\mathcal{N} = \{\neg A \sqcup B,\ \neg B \sqcup C,\ \neg B \sqcup \exists r.B\}$.

For $x \in \{\mathfrak{T}, \mathfrak{T}^u\}$, a x-derivation (tree) Δ built from a set of clauses $\mathcal{N}$ is a finite binary tree in which each leaf is labelled with a clause from $\mathcal{N}$ and each non-leaf node n is labelled with a clause $\mathcal{C}$ such that $\mathcal{C}$ results from an x-inference on the parent(s) of n in Δ. We say that Δ is a derivation of a clause $\mathcal{C}$ if the root of Δ is labelled with $\mathcal{C}$. A derivation of the empty clause is called a *refutation*. Every path $n_1, \ldots, n_m$ of nodes in Δ where n_1 is a leaf node and n_m is the root node induces an *inference path* $\alpha_2, \ldots, \alpha_m$, where $\alpha_i \in \mathsf{N_C} \cup \{\bot\}$ $(2 \leq i \leq m)$ denotes the concept name, or $\bot$, which has been resolved upon to obtain the clause that is the label of the node n_i. For a signature $\Upsilon \subseteq \mathsf{N_C}$ and a strict total order $\succ \subseteq \Upsilon \times \Upsilon$, a derivation Δ is a $(x, \Upsilon, \succ)$-derivation if for every inference path $\alpha_1, \ldots, \alpha_n$ of Δ (with $\alpha_i \in \mathsf{N_C} \cup \{\bot\}$ for every $1 \leq i \leq n$) there exists $0 \leq k \leq n$ such that $\{\alpha_1, \ldots, \alpha_k\} \subseteq \Upsilon$, $\alpha_j \succ \alpha_{j+1}$ or $\alpha_j = \alpha_{j+1}$ for every $1 \leq j < k$, and $\alpha_j \notin \Upsilon$ for every $k < j \leq n$.

We prove that for every unsatisfiable set of initial clauses there always exists a $(\mathfrak{T}, \Upsilon, \succ)$-refutation by extending the results and proof methods of Herzig and Mengin (2008).

Theorem 6 ($\mathfrak{T}$-Completeness)**.** *Let $\Upsilon \subseteq \mathsf{N_C}$, let $\succ \subseteq \Upsilon \times \Upsilon$ be a strict total order on Υ and let C and D be $\mathcal{ALC}$ concepts. Then it holds that $\models C \sqsubseteq D$ iff there exists a $(\mathfrak{T}, \Upsilon, \succ)$-derivation of the empty clause from the initial clauses* $\mathrm{Cls}(C \sqcap \neg D)$.

A weaker version of this result, stating that any derivation in $\mathfrak{T}$ can be reordered so that inferences on concept names from Υ always precede inferences on other concept names, or $\bot$, has been previously announced by Herzig and Mengin (2008); however, as we show in the full version of the paper, the proof appears to have some gaps.

To prove completeness for $\mathfrak{T}^u$, we observe the following link between derivations in $\mathfrak{T}$ and $\mathfrak{T}^u$. Let $\mathcal{N}$ be a set of clauses and let

$$\begin{aligned} \mathrm{Univ}_0(\mathcal{N}) &= \mathcal{N}; \\ \mathrm{Univ}_{i+1}(\mathcal{N}) &= \mathrm{Univ}_i(\mathcal{N}) \cup \\ &\qquad \textstyle\bigcup_{r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{N})} \{\, \forall r.\mathcal{C} \mid \mathcal{C} \in \mathrm{Univ}_i(\mathcal{N}) \,\} \\ \text{and } \mathrm{Univ}(\mathcal{N}) &= \textstyle\bigcup_{i \geq 0} \mathrm{Univ}_i(\mathcal{N}). \end{aligned}$$

Theorem 7. *Let $\mathcal{M}$ be a set of initial clauses and let $\mathcal{N}$ be a set of universal clauses. Additionally, let Δ be a $(\mathfrak{T}, \Upsilon, \succ)$-refutation from $\mathcal{M} \cup \mathrm{Univ}(\mathcal{N})$ such that there exists $n \in \mathbb{N}$ with $\mathrm{Depth}(\mathcal{C}) \leq n$ for every $\mathcal{C} \in$ Clauses(Δ). Then there exists a $(\mathfrak{T}^u, \Upsilon, \succ)$-derivation Δ^u of the empty clause from $\mathcal{M} \cup \mathcal{N}$ such that $\mathrm{Depth}(\mathcal{C}) \leq n$ for every $\mathcal{C} \in$ Clauses(Δ^u).*

We then use Theorems 6 and 7 and the fact that every $\mathcal{ALC}$-TBox can be internalised. Notice that the actual TBox internalisation $C_\mathcal{T}$ does not have to be computed as it is only used for the proof of completeness.

Corollary 8 ($\mathfrak{T}^u$-Completeness)**.** *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox, let $\Upsilon \subseteq \mathsf{N_C}$, let $\succ \subseteq \Upsilon \times \Upsilon$ be a strict total order on Υ and let C and D be $\mathcal{ALC}$ concepts. Then it holds that $\mathcal{T} \models C \sqsubseteq D$ iff there exists a $(\mathfrak{T}^u, \Upsilon, \succ)$-derivation of the empty clause from the universal clauses* $\mathrm{Cls}(\mathcal{T})$ *and the initial clauses* $\mathrm{Cls}(C \sqcap \neg D)$.

Computing Uniform Interpolants. The procedure UNIFORMINTERPOLANT depicted in Algorithm 1 takes as input an $\mathcal{ALC}$-TBox $\mathcal{T}$, a signature $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ such that $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$ and a strict total order $\succ \subseteq \Upsilon \times \Upsilon$ over $\Upsilon = \mathsf{sig}(\mathcal{T}) \setminus \Sigma$. Following the outline given by Herzig and Mengin (2008), after the clausification of $\mathcal{T}$, the procedure iterates over the concept names contained in Υ in descending order according to the relation $\succ$. In each iteration the clause set $\mathcal{N}$ is expanded with all possible $\mathfrak{T}^u$-inferences on the current concept name $A \in \Upsilon$. Finally, after iterating over all the concept names from $\Upsilon = \mathsf{sig}(\mathcal{T}) \setminus \Sigma$, the operator 'Supp' is applied on the resulting clauses, which replaces all occurrences of Υ concept names in clauses with $\top$ and then simplifies the resulting CNF.

Example 9. *For the clauses obtained in Example 5,* $\mathrm{Supp}(\{B\}, \neg A \sqcup C) = \neg A \sqcup C$, $\mathrm{Supp}(\{B\}, \neg A \sqcup \exists r.B) = \neg A \sqcup \exists r.\top$, $\mathrm{Supp}(\{B\}, \neg A \sqcup \exists r.(B, C)) = \neg A \sqcup \exists r.C$.

Algorithm 1

1: **procedure** UNIFORMINTERPOLANT($\mathcal{T}$, Σ, $\succ$)
2: $\quad\Upsilon := \mathsf{sig}(\mathcal{T}) \setminus \Sigma$
3: $\quad\mathcal{N} := \mathrm{Cls}(\mathcal{T})$
4: $\quad$**while** $\Upsilon \neq \emptyset$ **do**
5: $\quad\quad A := \max_{\succ}(\Upsilon)$
6: $\quad\quad\mathcal{N} := \mathrm{Res}^{\infty}_{\mathfrak{T}^u,\{A\}}(\mathcal{N})$
7: $\quad\quad\Upsilon := \Upsilon \setminus \{A\}$
8: $\quad$**end while**
9: $\quad$**return** $\mathcal{F}_{\Sigma}(\mathcal{T}) = \mathrm{Supp}(\mathsf{sig}(\mathcal{T}) \setminus \Sigma, \mathcal{N})$
10: **end procedure**

One can show that if Algorithm 1 terminates, then for all $\mathcal{ALC}$ Σ-concepts C, D such that there exists a $(\mathfrak{T}^u, \Upsilon, \succ)$-refutation Δ^u from the universal clauses $\mathrm{Cls}(\mathcal{T})$ and the initial clauses $\mathrm{Cls}(C \sqcap \neg D)$ it holds that $\mathcal{F}_{\Sigma}(\mathcal{T}) \models C \sqsubseteq D$. Thus, it follows from Corollary 8 that if Algorithm 1 terminates, it computes a Σ-uniform interpolant of $\mathcal{T}$. However, Algorithm 1 does not terminate if a uniform interpolant does not exist. For example, when applied to $\mathcal{T}$ from Example 2, Algorithm 1 can generate, among others, the infinite sequence of universal clauses $\neg A \sqcup \exists r.C$, $\neg A \sqcup \exists r.(C, \exists r.C), \ldots$ and so on. Moreover, as the TBox $\mathcal{T}$ from Example 2 is a subset of $\mathcal{T}'$, and so $\mathrm{Cls}(\mathcal{T}) \subseteq \mathrm{Cls}(\mathcal{T}')$, Algorithm 1 will derive, among others, the same clauses when it is applied on $\mathcal{T}'$. Thus, in some cases Algorithm 1 does not terminate even though a uniform interpolant exists.

To guarantee termination on all inputs, we focus on the notion of depth-bounded uniform interpolation (related to the notion of 'bounded forgetting' (Zhou and Zhang 2011)). Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox and let $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ be a signature. We say that an $\mathcal{ALC}$-TBox $\mathcal{T}_{\Sigma}$ is a *depth n-bounded uniform interpolant* of the TBox $\mathcal{T}$ w.r.t. Σ iff $\mathsf{sig}(\mathcal{T}_{\Sigma}) \subseteq \Sigma$, $\mathcal{T} \models \mathcal{T}_{\Sigma}$, and for every $\mathcal{ALC}$ Σ-concept inclusion $C \sqsubseteq D$ with $\mathcal{T} \models C \sqsubseteq D$ and $\max\{\mathrm{Depth}(C), \mathrm{Depth}(D)\} \leq n$ it holds that $\mathcal{T}_{\Sigma} \models C \sqsubseteq D$. Let $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ be the outcome of Algorithm 1 where in Step 6 only clauses up to depth m are generated. The following example shows that it might be necessary to consider intermediate clauses of a depth $m > n$ in order to preserve all the Σ-consequences of depth n entailed by $\mathcal{T}$.

Example 10. *Let $\mathcal{T} = \{A \sqsubseteq \exists r.C,\ C \sqsubseteq \exists s.\top,\ \neg B \sqsubseteq \forall s.\bot\}$, $\Sigma = \{A, B, r, s\}$, $\Upsilon = \{C\}$ and $\succ = \emptyset$. Then every $(\mathfrak{T}^u, \Upsilon, \succ)$-refutation from the universal clauses $\mathrm{Cls}(\mathcal{T})$ and the initial clauses $\{A, \forall r.\neg B\}$ derives the clause $\neg A \sqcup \exists r.(C, \exists s.\top)$.*

We establish, however, that by choosing the maximal depth of derived clauses appropriately, the procedure depicted in Algorithm 1 computes uniform interpolants that preserve consequences up to a specified depth n.

Theorem 11. *Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox, $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ a signature such that $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$, and let $n \geq 0$. Set $m = n + 2^{|\mathsf{sub}(\mathrm{Cls}(\mathcal{T}))|+1} + \max\{\mathrm{Depth}(\mathcal{C}) \mid \mathcal{C} \in \mathrm{Cls}(\mathcal{T})\}$, where $\mathsf{sub}(\mathrm{Cls}(\mathcal{T}))$ is the set of subconcepts of $\mathrm{Cls}(\mathcal{T})$. Then it holds that $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ is a depth n-bounded uniform interpolant of the TBox $\mathcal{T}$ w.r.t. Σ.*

We can combine this result with the results of (Lutz and Wolter 2011): for any $\mathcal{ALC}$-TBox $\mathcal{T}$ and signature Σ, if a Σ-uniform interpolant of $\mathcal{T}$ exists, then there exists a uniform interpolant of depth bounded by $2^{2^{|\mathcal{T}|+1}} + 1$. Thus, if $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$, there exists m, which can be computed based on the bound in Theorem 11 and the results of (Lutz and Wolter 2011), such that $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ is a Σ-uniform interpolant of $\mathcal{T}$.

The bound in Theorem 11 can be significantly improved if the TBox is acyclic. For an acyclic $\mathcal{ALC}$-TBox $\mathcal{T}$ we define $\mathrm{ExpansionDepth}(\mathcal{T}) = \max\{\mathrm{Depth}(A[\mathcal{T}]) \mid A \in \mathsf{sig}(\mathcal{T})\}$, where $A[\mathcal{T}]$ denotes the concept obtained by exhaustively replacing every concept B with C_B if $B \sqsubseteq C_B \in \mathcal{T}$ or $B \equiv C_B \in \mathcal{T}$.

Theorem 12. *Let $\mathcal{T}$ be an acyclic $\mathcal{ALC}$ TBox, $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ a signature such that $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$, and let $n \geq 0$. Set $m = \mathrm{ExpansionDepth}(\mathcal{T}) + n$. Then it holds that $\mathcal{F}_{\Sigma,m}(\mathcal{T})$ is a uniform interpolant limited to consequence depth n of the TBox $\mathcal{T}$ w.r.t. Σ.*

Note that in the description logic $\mathcal{EL}$ (i.e. the fragment of $\mathcal{ALC}$ that does not allow $\bot$, negation, disjunction, or universal quantification) the acyclicity of a TBox guarantees the existence of uniform interpolants (Konev, Walther, and Wolter 2009) for any signature Σ. Interestingly, this is not true in the case of $\mathcal{ALC}$. Moreover, as the following example shows, there exists an acyclic $\mathcal{EL}$-TBox $\mathcal{T}$ and a signature Σ for which no $\mathcal{ALC}$ Σ-uniform interpolant exists.

Example 13. *Consider $\Sigma = \{A, A_0, A_1, A_2, E, r\}$ and $\mathcal{T} = \{A \sqsubseteq \exists r.B,\ A_0 \sqsubseteq \exists r.(A_1 \sqcap B),\ E \equiv A_1 \sqcap B \sqcap \exists r.(A_2 \sqcap B)\}$. Then for every $n \geq 0$, $\mathcal{T}$ entails the inclusion*

$$A_0 \sqcap \prod_{i=1}^{n} \underbrace{\forall r. \ldots \forall r.}_{i}(A \sqcap \neg E \sqcap (A_1 \sqcup A_2)) \sqsubseteq \underbrace{\exists r. \ldots \exists r.}_{n} A_1.$$

This infinite sequence of $\mathcal{ALC}$ consequences of $\mathcal{T}$ cannot be captured by any $\mathcal{ALC}$ Σ-TBox $\mathcal{T}'$, which can be proved formally using Theorem 9 of Lutz and Wolter (2011).

Case Study

We have implemented a prototype of an inference computation architecture using the calculus $\mathfrak{T}^u$ and the inference relation $\Rightarrow_{\alpha}$ in Java. However, it turned out that our initial implementation of Algorithm 1 did not perform well in practice. This was in particular due to the fact that clauses can contain sets $\mathcal{F}$ of other clauses in existential literals $\exists r.\mathcal{F}$, which renders all the possible inferences on clauses from $\mathcal{F}$ 'explicit'. For example, if we resolve the universal clause which just consists of the existential literal $\exists r.(A)$ with the universal clauses $\neg A \sqcup B_1, \ldots, \neg A \sqcup B_n$ on the concept name A, then not only the clauses $\exists r.(A, B_1)$, $\exists r.(A, B_2)$, $\exists r.(A, B_3)$,... could be derived but all clauses of the form $\exists r.(A, \mathcal{G})$, where $\mathcal{G}$ is a subset of $\{B_1, \ldots, B_n\}$.

A common technique to reduce the number of inferences that have to be made is to use forward- and backward deletion of subsumed clauses (Bachmair and Ganzinger 2001). However, it is known (Auffray, Enjalbert, and Hébrard 1990) that the subsumption lemma (stating that if a clause $\mathcal{E}$ results from an inference involving two clauses $\mathcal{C}$ and $\mathcal{D}$, and

	Uniform Interpolation				Forgetting					
	$\lvert\Sigma \cap \mathsf{N_C}\rvert = 5$		$\lvert\Sigma \cap \mathsf{N_C}\rvert = 10$		$\lvert\Upsilon\rvert = 10$		$\lvert\Upsilon\rvert = 15$		$\lvert\Upsilon\rvert = 25$	
	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms	Success Rate (%)	Avrg # Axioms
AMINO-ACID v1.2	100	61.40	92	143.35	100	645.67	87	665.24	64	396.98
BHO v0.4	71	30.01	16	52.43	99	2374.73	99	2363.42	91	2383.96
CAO v1.4	100	279.02	100	283.33	100	369.54	100	369.22	10	366.07
CDAO	100	288.21	100	288.42	100	293.48	100	293.41	10	293.02
CHEMBIO v1.1	92	7 1.89	60	94.40	100	295.85	100	293.09	10	293.64
CPRO v0.85	100	585.08	100	533.82	100	307.76	100	309.46	10	316.31
DDI v0.9	100	249.80	100	259.41	100	276.27	100	278.55	10	276.61
DIKB v1.4	2	1591.50	0	-	97	622.67	83	689.44	56	816.39
GRO v0.5	0	-	0	-	94	959.85	91	940.03	79	997.59
IDO	0	-	0	-	94	1202.71	90	1203.78	80	1215.36
LIPRO v1.1	73	7.93	58	13.22	91	2287.24	58	2381.43	45	2297.37
NCI v08.10e	23	887.34	1	1397.00	97	100693.26	98	100611.60	99	100889.50
NEOMARK v4.1	31	19.45	14	27.28	100	338.52	100	333.26	10	324.86
OMRSE	100	485.00	100	485.00	100	485.00	100	485.00	10	485.00
OBIWS v1.1	100	112.56	100	118.70	100	189.66	100	187.71	10	184.13
ONTODM v1.1.1	0	-	0	-	98	1711.40	98	1704.67	93	1693.61
OPL	100	829.41	100	832.93	100	848.60	100	848.99	10	848.73
PROPREO v1.1	41	2.07	19	31.84	100	561.43	100	560.85	99	578.08
RNAO r113	100	355.86	100	362.83	100	439.64	100	439.10	10	439.71
SAO v1.2.4	0	-	0	-	99	2702.23	100	2700.85	98	2715.30
SITBAC v1.3	0	-	0	-	93	508.40	93	537.48	79	595.51
TOK v0.2.1	0	-	0	-	97	496.12	93	529.06	72	567.11
VSO	0	-	0	-	83	348.87	79	397.65	50	371.38

Table 1: Uniform Interpolation and Forgetting for BioPortal Ontologies on Small Signatures.

if there exist clauses $\mathcal{C}'$, $\mathcal{D}'$ such that $\mathcal{C}'$ subsumes $\mathcal{C}$ and $\mathcal{D}'$ subsumes $\mathcal{D}$, then either $\mathcal{E}$ is subsumed by one of $\mathcal{C}'$, $\mathcal{D}'$, or a clause $\mathcal{E}'$ can be derived from $\mathcal{C}'$ and $\mathcal{D}'$ such that $\mathcal{E}'$ subsumes $\mathcal{E}$) does not hold even in the modal logic K for the standard *minimal subsumption relation* $\leq_s$ (Auffray, Enjalbert, and Hébrard 1990) and $\Rightarrow_\alpha$. To be able to prove that one can safely discard subsumed clauses, we have modified the inference relation $\Rightarrow_\alpha$ by introducing the following additional rule (rule $\exists_f$)

$$\mathcal{C}_1 \sqcup \forall r.\mathcal{D},\ \ \mathcal{C}_2 \sqcup \exists r.\mathcal{F} \Longrightarrow_{\exists_f} \mathcal{C}_1 \sqcup \mathcal{C}_2 \sqcup \exists r.(\mathcal{F},\mathcal{D}).$$

We will denote the resulting inference relation by $\Rightarrow_\alpha^f$ with $\alpha \in \mathsf{N_C} \cup \{\bot, \exists_f\}$. One can then prove that a variant of the subsumption lemma holds for the relations $\leq_s$ and $\Rightarrow_\alpha^f$, which allows us to employ forward- and backward deletion of subsumed clauses in our implementation.

In order to further speed up computations, we first extract the locality-based $\top\bot^*$ Σ-module (Cuenca Grau et al. 2008; Sattler, Schneider, and Zakharyaschev 2009) for a given TBox $\mathcal{T}$. The locality-based module entails the same Σ-inclusions as the TBox $\mathcal{T}$ but it is often considerably smaller in size. We also rely on ontologies to have structure: if a concept name occurs in several inclusions, it is likely that it occurs in the same syntactic pattern. We therefore transform clause sets as follows.

1. If the clause set contains some clauses $\mathcal{C}_1 \sqcup \mathcal{D}_\Upsilon, \ldots, \mathcal{C}_m \sqcup \mathcal{D}_\Upsilon$ such that for every $1 \leq i \leq m$ we have $\mathsf{sig}(\mathcal{C}_i) \cap \Upsilon = \emptyset$, we rewrite them into $X \sqcup \mathcal{D}_\Upsilon$, where $X \equiv \mathcal{C}_1 \sqcap \ldots \sqcap \mathcal{C}_m$, perform forgetting on Υ symbols and then replace X with its definition.
2. If the clause set contains a clause $\mathcal{C} \sqcup \exists r.(\mathcal{F}_\Upsilon, \mathcal{G}_1) \sqcup \ldots \sqcup \exists r.(\mathcal{F}_\Upsilon, \mathcal{G}_m)$ where $\mathsf{sig}(\mathcal{G}_i) \cap \Upsilon = \emptyset$ for every $1 \leq i \leq m$, we rewrite it into $\mathcal{C} \sqcup \exists r.(\mathcal{F}_\Upsilon, Y)$, where $Y \equiv \mathcal{G}_1 \sqcup \ldots \sqcup \mathcal{G}_m$, perform forgetting on Υ and then replace Y with its definition.

Experimental setting. All experiments were conducted on PCs equipped with an Intel Core i5-2500K CPU running at 3.30GHz. 15 GiB of RAM were allocated to the Java VM and an execution timeout of 60 CPU minutes was imposed on each problem. Whenever necessary we pre-processed the ontologies we used for our experiments as follows. For a given ontology $\mathcal{T}$, we first rewrote concept disjointness statements and role domain/range restrictions into $\mathcal{ALC}$ inclusions and then removed any remaining axiom which contained non-$\mathcal{ALC}$ concept (or role) constructors to obtain the $\mathcal{ALC}$-fragment of $\mathcal{T}$. We used Algorithm 1 to forget concept names one by one i.e. for $\Upsilon = \{A_1, \ldots, A_n\}$, Algorithm 1 was applied iteratively on $A_1, \ldots, A_n$, and we did *not* impose a bound on the depth of clauses; so the computed clause

	Computing $\mathsf{Diff}_\Sigma(\mathcal{T}_{i+1}, \mathcal{T}_i)$			Computing $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$		
	Successful/ Total Runs	Success Rate (%)	Average # of Witnesses	Successful/ Total Runs	Success Rate (%)	Average # of Witnesses
BDO	3/5	60	12.33	5/5	100	211.40
CHEMINF	25/26	96	7.00	26/26	100	2.26
COGAT	4/4	100	272.00	3/4	75	4.00
JERM	8/13	61	7.00	9/13	69	9.33
NCI	101/108	93	787.10	105/108	97	906.20
NEMO	14/15	93	13.35	15/15	100	33.46
NPO	12/18	66	27.08	12/18	66	5.58
OMRSE	11/11	100	0.54	11/11	100	0.00
OPL	4/4	100	18.75	4/4	100	2.25
SIO	18/35	51	0.00	19/35	54	0.00

Table 3: Computing the Logical Difference between Ontology Versions on their Common Signature.

	$\lvert\Sigma \cap \mathsf{N_C}\rvert$	Success Rate (%)	Avrg # Axioms
DIKB	5	85	7.482
	10	60	14.033
	15	44	25.114
NCI	5	82	1.62
	10	64	2.65
	50	65	21.369
	100	56	41.089
	150	41	63.146

Table 2: Computing Uniform Interpolants of DIKB v1.4 and of NCI v08.10e Limited to Expansion Depth 3.

sets contain depth n-bounded uniform interpolants for every $n > 0$. Thus, in all the experiments reported on we computed *true* Σ*-uniform interpolants* (i.e. not a depth-bounded variant). The correctness of our extensions to Algorithm 1 can be shown by model-theoretic arguments.

Experiments with small signatures. We applied our uniform interpolation tool to compute uniform interpolants w.r.t. small concept signatures $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ with $\mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R} = \Sigma \cap \mathsf{N_R}$ for 21 small to medium size ontologies taken from the BioPortal repository[2]. The number of axioms in the selected ontologies ranges from 192 (for the *Ontology of Medically Related Social Entries*) to 2702 (for the *Subcellular Anatomy Ontology*). To make the experiments more interesting, we also included version 08.10e of the *National Cancer Institute Thesaurus* (NCI). For each considered sample size x and terminology $\mathcal{T}$ we generated 100 signatures Σ by randomly choosing x concept names from $\mathsf{sig}(\mathcal{T})$ and by adding all the role names from $\mathsf{sig}(\mathcal{T})$ to Σ. The results that we obtained are shown in Table 1.

In the left half of Table 1 one can see that the number of successful computations decreased while the size of $\Sigma \cap \mathsf{N_C}$ was increasing, which seems to be due to the fact that the $\top\bot^*$ Σ-modules then contain more symbols that lead to a large number of inferences. Most uniform interpolants that we obtained are relatively small and contain a lot of expressions of the form $\exists r_1 \ldots \exists r_n.\top$. In some cases the process of forgetting certain intermediate concept names generated a few hundred clauses that were simplified or deleted in the remaining computation steps. The success rate, however, varied significantly from one ontology to another. To further investigate this phenomenon, we computed uniform interpolants for a *fragment* of version 08.10e of NCI and for a fragment of version 1.4 of the *Drug Interaction Knowledge Base* (DIKB) that are of expansion depth 3 (that is, we removed all the axioms from both ontologies that led to an expansion depth greater than 3). The resulting DIKB fragment is a small acyclic terminology that contains 120 concept names, 27 roles names, and 127 axioms. The NCI fragment is also an acyclic terminology with 53571 concept names, 78 role names and 62494 axioms (of which 2362 are of the form $A \equiv C$). The results obtained are shown in Table 2. Limiting the expansion depth drastically improved the performance of our prototype implementation with the success rate for signatures containing 5 randomly selected concept names rising from 2% to 85% in the case of DIKB and from 23% to 82% in the case of NCI. For NCI our tool was capable of handling signatures containing up to 150 randomly selected concept names.

As proof of concept for ontology obfuscation, we applied our uniform interpolation tool on (a fragment of) the *Lipid Ontology* (LIPRO) to forget 45 concept names which are intermediate concept names in the ontology's induced concept hierarchy, i.e. those concept names group certain subconcepts together to give structure to the ontology. LIPRO is an acyclic terminology with 593 axioms, 574 concept names and one role name. The maximal size of an axiom is 50. It then took 192 CPU seconds to compute the uniform interpolant, which contains 3415 axioms that have a maximal size of 283. The uniform interpolant that we computed thus approximately contains 6 times more axioms than the ori-

[2] All ontologies used for the experiments reported on in this section can be accessed from the BioPortal repository: `http://bioportal.bioontology.org/ontologies`

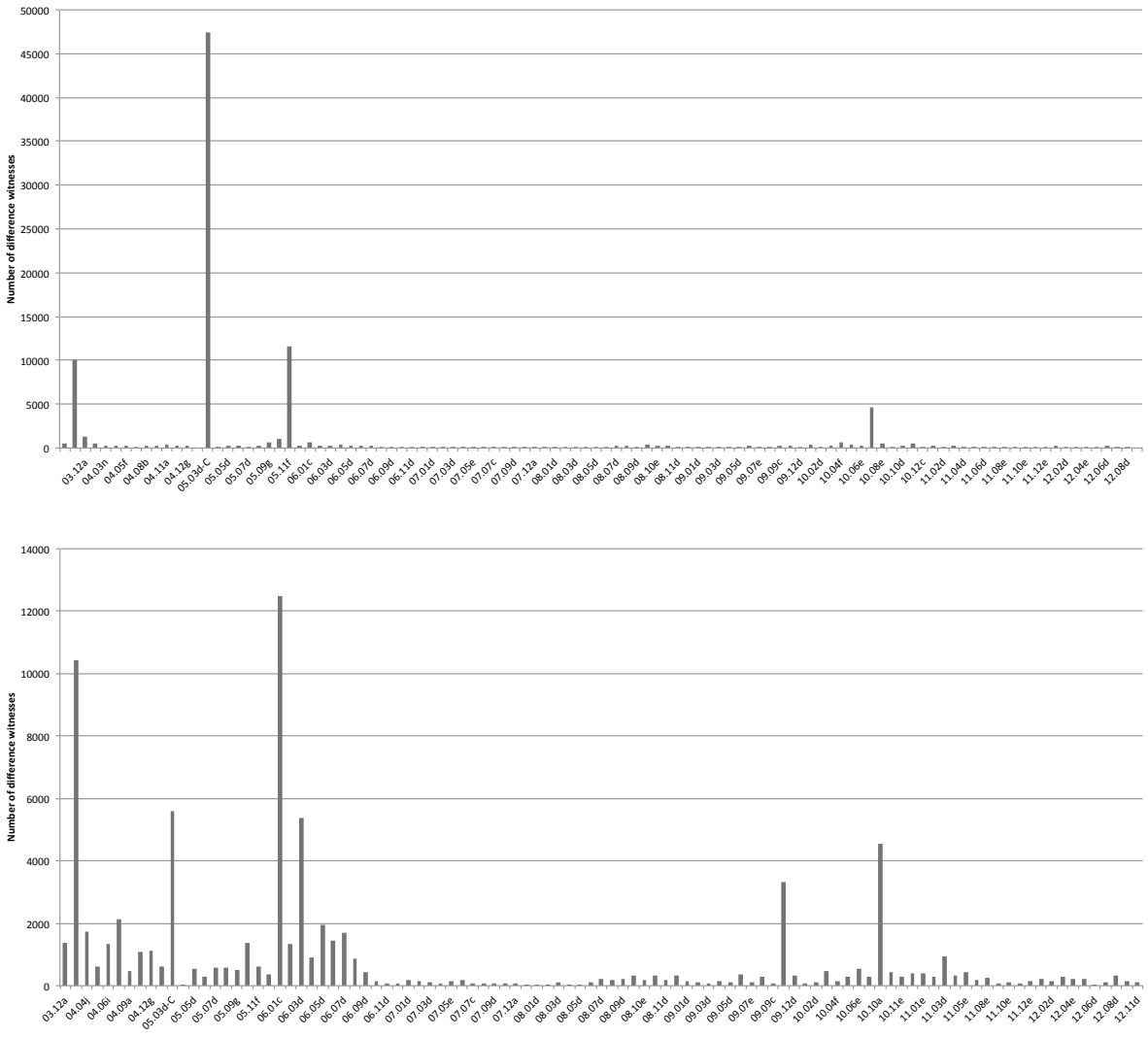

Figure 2: Logical Difference between NCI Versions i and $i+1$ (Top) and Versions $i+1$ and i (Bottom).

ginal ontology and the maximal axiom size has increased by a factor of 6 as well. Notice that most of the original structure of the ontology has been destroyed while preserving all the consequences entailed by the retained concept names.

Finally, in the right half of Table 1 we report on our success rate for forgetting a small number of concept names. Notice that our prototype implementation performs significantly better in this scenario. This observation suggests that our tool is suitable for checking whether a change made to an ontology interferes with the meaning of the terms outside the (typically small) fragment under consideration in the context of computing the logical difference between two versions of an ontology.

Computing the Logical Difference. We selected 10 ontologies that have at least 5 submissions and whose expressivity is at least $\mathcal{ALC}$, including 109 versions of the NCI Thesaurus, from the BioPortal repository.

For every pair of consecutive versions $\mathcal{T}_i$ and $\mathcal{T}_{i+1}$, where version $i+1$ represents the more recent version, and every considered signature Σ, we computed both $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$ and $\mathsf{Diff}_\Sigma(\mathcal{T}_{i+1}, \mathcal{T}_i)$. We used the reasoner FaCT++ v1.6.2 (Tsarkov and Horrocks 2006) to determine whether any axiom $C \sqsubseteq D \in \mathcal{T}_i^{(\Sigma)}$ is a witness of $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$, where $\mathcal{T}_i^{(\Sigma)}$ is the Σ-uniform interpolant of $\mathcal{T}_i$ computed with our tool (similarly for $\mathsf{Diff}_\Sigma(\mathcal{T}_{i+1}, \mathcal{T}_i)$). Note that the signatures Σ we used for the experiments contained (at least) all the roles present in $\mathcal{T}_i$ due to the limitations of our uniform interpolation procedure. However, such a restriction does not limit the applicability of logical difference in the sense that differences involving role names can still be detected. Note further that the results we obtained here are not directly comparable with the logical difference computed for description logics of the $\mathcal{EL}$ family (Konev, Walther, and Wolter 2008; Konev et al. 2012) as illustrated

$\lvert(\mathsf{sig}(\mathcal{T}) \setminus \Sigma) \cap \mathsf{N_C}\rvert$	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v08.09d}}, \mathrm{NCI}_{\text{v08.10e}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v05.03d}}, \mathrm{NCI}_{\text{v05.05d}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v05.12f}}, \mathrm{NCI}_{\text{v06.01c}})$ Success Rate (%)	Avrg # Witnesses
5	100	446.01	100	47 458.14	100	11 564.71
10	99	446.05	100	47 456.66	97	11 595.85
20	100	445.95	100	47 453.26	94	11 671.79
50	88	445.73	100	47 436.72	84	11 849.16
100	88	445.67	100	47 403.76	70	12 468.64

$\lvert(\mathsf{sig}(\mathcal{T}) \setminus \Sigma) \cap \mathsf{N_C}\rvert$	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v08.10e}}, \mathrm{NCI}_{\text{v08.09d}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v05.05d}}, \mathrm{NCI}_{\text{v05.03d}})$ Success Rate (%)	Avrg # Witnesses	$\mathsf{Diff}_\Sigma(\mathrm{NCI}_{\text{v06.01c}}, \mathrm{NCI}_{\text{v05.12f}})$ Success Rate (%)	Avrg # Witnesses
5	98	2338.89	96	1347.92	99	13 704.29
10	98	2338.45	98	1348.47	100	13 788.15
20	97	2347.08	95	1348.66	95	13 841.52
50	92	2340.72	86	1351.56	87	14 062.52
100	86	2385.88	74	1354.04	80	14 504.40

Table 4: Computing the Logical Difference between Versions of NCI.

by Example 13.

In our first experiment we used $\Sigma = (\mathsf{sig}(\mathcal{T}_i) \cap \mathsf{sig}(\mathcal{T}_{i+1})) \cup \mathsf{N_R}$. This test captures any change to the meaning of the terms common to both versions. The results of computing the logical difference are given in Table 3. Notice that the success rate of computing $\mathsf{Diff}_\Sigma(\mathcal{T}_i, \mathcal{T}_{i+1})$ was slightly higher than the one of the converse direction. This observation can probably be attributed to the fact that these cases correspond to knowledge contained in an older version being removed from a newer one, which does not seem to happen often.

Interestingly, we could observe one of the highest success rates among all our experiments whilst computing logical differences for distinct versions of NCI. This can possibly be explained by the fact that versions of NCI are released frequently and changes to the ontology are hence introduced gradually. Figure 2 depicts the number of witnesses that correspond to the logical difference between consecutive versions of NCI on their common signature. Gonçalves, Parsia, and Sattler (2012) provide a comprehensive analysis of the changes between 14 consecutive versions of NCI using various techniques, ranging from a manual inspection of the log files to approximations of the logical difference. Versions 05.12f, 06.01c, and 06.08d were identified as having the highest number of differences. In our experiments, the highest number of logical difference witnesses were also present in NCI version 06.01c; the computations for versions 05.12f and 06.08d did not finish in time.

Furthermore, to make the experiments more challenging for the reasoner, we focused on comparing version i with version $i + 1$, and vice versa, on the 2 pairs of NCI versions for which the highest number of difference witnesses was identified in the first experiment. We also included version 08.10e as this is the last acyclic $\mathcal{ALC}$ TBox in the corpus. We performed tests on randomly generated large signatures Σ with $\Sigma \cap \mathsf{N_R} = \mathsf{sig}(\mathcal{T}) \cap \mathsf{N_R}$. In that way the computed uniform interpolants remained rather large as well.

For each sample size $x \in \{5, 10, 20, 50, 100\}$ we generated 100 signatures by randomly choosing $|\mathsf{sig}(\mathcal{T}) \cap \mathsf{N_C}| - x$ concept names from $\mathsf{sig}(\mathcal{T})$ and by including all the role names from $\mathsf{sig}(\mathcal{T})$. The results that we obtained are now shown in Table 4.

One can observe that as the size of $\mathsf{sig}(\mathcal{T}) \setminus \Sigma$ increased, i.e. more symbols had to be forgotten from the $\top\bot^*$ Σ-modules, the success rate dropped slightly. Overall, the average number of witnesses and the average maximal size of the witnesses remained comparable throughout the different sample sizes. Also, the axioms generated by the computation of the uniform interpolant did not pose a problem for FaCT++ as computing the logical difference for a given signature never took more than 20 seconds in our experiments.

Conclusion

In this paper we presented an approach based on clausal resolution for computing uniform interpolants of $\mathcal{ALC}$-TBoxes $\mathcal{T}$ w.r.t. signatures $\Sigma \subseteq \mathsf{sig}(\mathcal{T})$ that contain all the role names present in $\mathcal{T}$. We proved that whenever the saturation process under $\mathcal{ALC}$-resolution terminates, the algorithm computes a uniform interpolant. To guarantee termination on all inputs, we introduced a depth-bounded version of our algorithm. We showed that by choosing an appropriate bound on the depth of clauses, one can axiomatise all Σ-inclusions implied by the given TBox up to a specified depth. Combined with a known bound on the size of uniform interpolants, our depth-bounded procedure always computes a uniform interpolant if it exists.

In the second part of this paper we investigated how often our unrestricted resolution-based algorithm terminates with a uniform interpolant by applying our prototype implementation on a number of case studies. Our findings suggest that despite a high computational complexity uniform interpolants can be computed in many practical cases. The com-

putation procedure could further benefit from better redundancy elimination techniques, which, together with extending our approach to forgetting role names, constitutes future work. It would also be interesting to explore proof strategies for our resolution calculi that guarantee termination when uniform interpolants exist.

References

Auffray, Y.; Enjalbert, P.; and Hébrard, J.-J. 1990. Strategies for modal resolution: Results and problems. *Journal of Automated Reasoning* 6(1):1–38.

Bachmair, L., and Ganzinger, H. 2001. Resolution theorem proving. In *Handbook of automated reasoning*, volume 1. Elsevier. chapter 2, 19–99.

Collberg, C. S.; Thomborson, C. D.; and Low, D. 1998. Manufacturing cheap, resilient, and stealthy opaque constructs. In *Proceedings of POPL '98*, 184–196. ACM.

Conradi, R., and Westfechtel, B. 1998. Version models for software configuration management. *ACM Computing Surveys (CSUR)* 30(2):232–282.

Cuenca Grau, B.; Horrocks, I.; Kazakov, Y.; and Sattler, U. 2008. Modular reuse of ontologies: theory and practice. *Journal of Artificial Intelligence Research (JAIR)* 31:273–318.

Eiter, T.; Ianni, G.; Schindlauer, R.; Tompits, H.; and Wang, K. 2006. Forgetting in managing rules and ontologies. In *Proceedings of the 2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006)*, 411–419. IEEE Computer Society.

Gonçalves, R. S.; Parsia, B.; and Sattler, U. 2012. Concept-based semantic difference in expressive description logics. In *Proceedings of ISWC 2012*, volume 7649 of *LNCS*, 99–115. Springer.

Herzig, A., and Mengin, J. 2008. Uniform interpolation by resolution in modal logic. In *Proceedings of JELIA 2008*, volume 5293 of *LNCS*, 219–231. Springer.

Jiménez-Ruiz, E.; Grau, B. C.; Horrocks, I.; and Llavori, R. B. 2009. ContentCVS: A CVS-based collaborative ontology engineering tool. In *Proceedings of the Workshop on Semantic Web Applications and Tools for Life Sciences*, volume 559 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Jiménez-Ruiz, E.; Cuenca Grau, B.; Horrocks, I.; and Llavori, R. B. 2011. Supporting concurrent ontology development: Framework, algorithms and tool. *Data & Knowledge Engineering* 70(1):146–164.

Klein, M. C. A.; Fensel, D.; Kiryakov, A.; and Ognyanov, D. 2002. Ontology versioning and change detection on the web. In *Proceedings of EKAW 2002*, volume 2473 of *LNCS*. Springer. 247–259.

Konev, B.; Ludwig, M.; Walther, D.; and Wolter, F. 2012. The logical difference for the lightweight description logic $\mathcal{EL}$. *Journal of Artificial Intelligence Research (JAIR)* 44:633–708.

Konev, B.; Walther, D.; and Wolter, F. 2008. The logical difference problem for description logic terminologies. In *Proceedings of IJCAR 2008*, volume 5195 of *LNCS*, 259–274. Springer.

Konev, B.; Walther, D.; and Wolter, F. 2009. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Proceedings of IJCAI 2009*, 830–835.

Koopmann, P., and Schmidt, R. A. 2013. Uniform interpolation of $\mathcal{ALC}$-ontologies using fixpoints. In *Proceedings of FroCoS 2013*, volume 8152 of *LNCS*. Springer.

Ludwig, M., and Konev, B. 2013. Towards practical uniform interpolation and forgetting for $\mathcal{ALC}$ TBoxes. In *Proceedings of DL 2013*, volume 1014 of *CEUR Workshop Proceedings*, 377–389. CEUR-WS.org.

Lutz, C., and Wolter, F. 2011. Foundations for uniform interpolation and forgetting in expressive description logics. In *Proceedings of IJCAI 2011*, 989–995.

Lutz, C.; Seylan, I.; and Wolter, F. 2012. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proceedings of KR 2012*. AAAI Press.

Nikitina, N., and Rudolph, S. 2012. ExpExpExplosion: Uniform interpolation in general $\mathcal{EL}$ terminologies. In *Proceedings of ECAI 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, 618–623. IOS Press.

Noy, N. F., and Musen, M. A. 2002. PromptDiff: A fixed-point algorithm for comparing ontology versions. In *Proceedings of AAAI 2002*, 744–750. AAAI Press.

Reiter, R., and Lin, F. 1994. Forget it! In *Proceedings of AAAI Fall Symposium on Relevance*, 154–159. AAAI Press.

Sattler, U.; Schneider, T.; and Zakharyaschev, M. 2009. Which kind of module should I extract? In *Proceedings of DL 2009*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org.

ten Cate, B.; Conradie, W.; Marx, M.; and Venema, Y. 2006. Definitorially complete description logics. In *Proceedings of KR 2006*, 79–89. AAAI Press.

Tsarkov, D., and Horrocks, I. 2006. FaCT++ description logic reasoner: System description. In *Proceedings of IJCAR 2006*, volume 4130 of *LNCS*, 292–297. Springer.

Wang, Z.; Wang, K.; Topor, R.; and Pan, J. Z. 2008. Forgetting concepts in DL-Lite. In *Proceedings of ESWC2008*, volume 5021 of *LNCS*, 245–257. Springer.

Wang, Z.; Wang, K.; Topor, R. W.; and Zhang, X. 2010. Tableau-based forgetting in $\mathcal{ALC}$ ontologies. In *Proceedings of ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 47–52. IOS Press.

Wang, K.; Wang, Z.; Topor, R.; Pan, J. Z.; and Antoniou, G. 2012. Eliminating concepts and roles from ontologies in expressive descriptive logics. *Computational Intelligence*. DOI: 10.1111/j.1467-8640.2012.00442.x.

Zhou, Y., and Zhang, Y. 2011. Bounded forgetting. In *Proceedings of AAAI 2011*. AAAI Press.